

Probabilistic Programming Homework 3

Deadline: 6:00pm on 29 May 2020 (Friday)

Submit your solutions in KLMS by 6:00pm on 29 May 2020 (Friday). (Reminder: We adopt a very strict policy for handling dishonest behaviours. If a student is found to copy answers from fellow students or other sources in his or her homework submission, she or he will get F.)

Question 1

This question is concerned with operational semantics for probabilistic programs. Throughout the question, we will use the notions of expression, value, redex and evaluation context, and the \Rightarrow and \rightsquigarrow relations for the likelihood weighted importance sampler that we used in the course. Here is a reminder of these notions:

Expression $e ::= c \mid x \mid (\text{fn } [x_1 \dots x_n] e) \mid (e_0 e_1 \dots e_n) \mid (\text{if } e_0 e_1 e_2)$

Constant $c ::= \text{true} \mid \text{false} \mid \text{and} \mid \text{not} \mid \leq \mid 1.2 \mid + \mid * \mid \dots$
 $\mid d \mid \text{sample} \mid \text{observe}$

Distribution Constructor $d ::= \text{normal} \mid \text{flip} \mid \text{poisson} \mid \dots$

Value $f ::= c \mid x \mid (\text{fn } [x_1 \dots x_n] e) \mid (d v_1 \dots v_n)$

Redex $f ::= (v_0 v_1 \dots v_n) \quad (v_0 \text{ is not a dist. constructor})$
 $\mid (\text{if } v e_1 e_2)$

Evaluation Context $C ::= [-] \mid (v_0 v_1 \dots v_{m-1} C e_{m+1} \dots e_{m+n}) \mid (\text{if } C e_1 e_2)$

We also recall the \implies and \rightsquigarrow relations for the likelihood weighted importance sampler:

$$\begin{array}{c}
\overline{(\text{if true } e_1 \ e_2), w \implies e_1, w} \\
\\
\frac{v \neq \text{true}}{(\text{if } v \ e_1 \ e_2), w \implies e_2, w} \\
\\
\overline{((\text{fn } [x_1 \dots x_n] \ e) \ v_1 \dots v_n), w \implies e[x_1 := v_1, \dots, x_n := v_n], w} \\
\\
\frac{c \notin \{\text{sample}, \text{observe}\} \quad c \text{ not dist. constructor} \quad c' \leftarrow \text{compute_op}(c, [v_1, \dots, v_n])}{(c \ v_1 \dots v_n), w \implies c', w} \\
\\
\frac{c' \leftarrow \text{sample_dist}(d, [v_1, \dots, v_n])}{(\text{sample } (d \ v_1 \dots v_n)), w \implies c', w} \quad \frac{w' \leftarrow \text{score_dist}(d, [v_1, \dots, v_n], c)}{(\text{observe } (d \ v_1 \dots v_n) \ c), w \implies c, w \cdot w'} \\
\\
\frac{(v_0 \ v_1 \dots v_n) \text{ is not one of the cases from above}}{(v_0 \ v_1 \dots v_n), w \implies \text{nil}, w} \\
\\
\frac{f, w \implies e', w'}{C[f], w \rightsquigarrow C[e'], w'}
\end{array}$$

You can find more information about them in the handwritten notes in the course webpage. Also, the note “Implementing Inference Algorithm for Probabilistic Programs” in the course webpage contains essentially the same information but with slightly different notations and presentation.

- (a) Typically, the `let` construct in a programming language is introduced as a macro that gets expanded as follows:

$$(\text{let } [x \ e] \ e') \equiv ((\text{fn } [x] \ e') \ e).$$

Suppose that we do not want to treat `let` as macro, but we want to regard it as one of the core constructs in the language. Extend the definitions of redex f , evaluation context C , and the \implies relation such that `let` expressions are evaluated directly by the \implies relation, instead of first being macro-expanded and then being evaluated. Hint: You need to add one new case to the current definitions of redex f and evaluation context C , and one new rule for \implies . (15 marks)

- (b) Consider the following program:

```

(let [x (sample (uniform-continuous 0.1 0.9))]
  (let [y (observe (flip x) true)]
    (if (> x 0.5) 0 1)))

```

Let us use e to denote this program. There are many ways of evaluating $(e, 1)$ to a value-weight pair using the \rightsquigarrow relation extended in Part (a). For instance, the following repeated

application of the \rightsquigarrow relation leads to the pair of value 1 and weight 0.2:

$$\begin{aligned}
& \left(\begin{array}{l} (\text{let } [x \text{ (sample (uniform-continuous 0.1 0.9))}] \\ (\text{let } [y \text{ (observe (flip } x) \text{ true})}] \\ (\text{if } (> \ x \ 0.5) \ 0 \ 1))) \end{array} , 1 \right) \\
& \rightsquigarrow \left(\begin{array}{l} (\text{let } [x \ 0.2] \\ (\text{let } [y \text{ (observe (flip } x) \text{ true})}] , 1 \\ (\text{if } (> \ x \ 0.5) \ 0 \ 1))) \end{array} \right) \\
& \rightsquigarrow \left(\begin{array}{l} (\text{let } [y \text{ (observe (flip 0.2) true)] \\ (\text{if } (> \ 0.2 \ 0.5) \ 0 \ 1)) \end{array} , 1 \right) \rightsquigarrow \left(\begin{array}{l} (\text{let } [y \text{ true}] \\ (\text{if } (> \ 0.2 \ 0.5) \ 0 \ 1)) \end{array} , 0.2 \right) \\
& \rightsquigarrow ((\text{if } (> \ 0.2 \ 0.5) \ 0 \ 1), 0.2) \rightsquigarrow ((\text{if false } 0 \ 1), 0.2) \rightsquigarrow (1, 0.2)
\end{aligned}$$

(Here I used an unexplained rule of \rightsquigarrow for `let`) Using your rule for `let`, give two other possible execution sequences of the program e . Make sure that the final values of your execution sequences are 0 and 1, respectively. (15 marks)

Question 2

The lightweight Metropolis-Hastings algorithm uses the following acceptance ratio:

$$\alpha((v_{n-1}, w_{n-1}, S_{n-1}), (v', w', S')) = \min \left(1, \frac{w' \cdot |S_{n-1}|}{w_{n-1} \cdot |S'|} \right).$$

Here $(v_{n-1}, w_{n-1}, S_{n-1})$ is the current state consisting of value v_{n-1} , weight w_{n-1} , and sequence of random choices S_{n-1} , and (v', w', S') is a newly proposed state. With the probability $\alpha((v_{n-1}, w_{n-1}, S_{n-1}), (v', w', S'))$, we accept this new state and set the next state (v_n, w_n, S_n) to be (v', w', S') . For detail, look at the note “Implementing Inference Algorithm for Probabilistic Programs” in the course webpage.

Suppose that instead of the correct acceptance ratio above, we used the following variant:

$$\alpha'((v_{n-1}, w_{n-1}, S_{n-1}), (v', w', S')) = \min \left(1, \frac{w'}{w_{n-1}} \right).$$

Explain why this variant computes a wrong estimate for the following program:

```
(if (sample (flip 0.5))
  (or (sample (flip 0.5)) true)
  false)
```

This (slightly silly) program does not contain any `observe`. Thus, its prior and posterior distributions coincide. In fact, the posterior distribution on boolean values assigns 0.5 to `true` and 0.5 to `false`. Hint: One way to answer this question is to focus on what the loop body of the variant of the lightweight MH algorithm does on v_n and to show that the target posterior distribution is not an invariant distribution (or stationary distribution) of the loop body. (30 marks)

Question 3

This question is about stochastic variational inference. Consider finite sets X and Y that are ranged over by x and y , respectively. Assume that we are given a probability distribution $p(x, y)$ on $X \times Y$ in the form of $p(x)$ and $p(y|x)$. The x part of (x, y) is an unobserved latent state, and the y part is an observation. Also, assume that we are given a particular observation y_0 , and that we are interested in approximating the posterior distribution $p(x|y_0)$.

In variational inference, we solve this posterior-inference problem as follows. We consider a proposal distribution $q_\theta(x)$ parameterised by $\theta \in \mathbb{R}^m$ such that for fixed x , the function $\theta \mapsto q_\theta(x)$ is a differentiable function from \mathbb{R}^m to the interval $[0, 1]$. Then, we set the following optimisation problem:

$$\operatorname{argmin}_\theta \operatorname{KL}\left[q_\theta(x) \parallel p(x|y_0)\right].$$

Finally, we solve this optimisation problem approximately.

Many variational-inference algorithms solve the above optimisation problem approximately by stochastic gradient descent. The core of such an algorithm often lies in its sample-based estimator of the following gradient:

$$\nabla_\theta \operatorname{KL}\left[q_\theta(x) \parallel p(x|y_0)\right]. \quad (1)$$

In the lectures, we covered one such estimator, called score estimator or REINFORCE.

- (a) Prove the following equation that forms the basis of the score estimator (and hence the basis of the stochastic variational-inference algorithm based on the score estimator as well).

$$\nabla_\theta \operatorname{KL}\left[q_\theta(x) \parallel p(x|y_0)\right] = \mathbb{E}_{x \sim q_\theta(x)} \left[\left(\nabla_\theta \log(q_\theta(x)) \right) \cdot \log \frac{q_\theta(x)}{p(x, y_0)} \right]. \quad (2)$$

(10 marks)

- (b) The equation in (2) gives rise to the following algorithm for estimating the gradient in (1). First, sample x_1, \dots, x_N from $q_\theta(x)$. Next, estimate the gradient by

$$\frac{1}{N} \cdot \sum_{i=1}^N \left(\nabla_\theta \log(q_\theta(x_i)) \right) \cdot \log \frac{q_\theta(x_i)}{p(x_i, y_0)}$$

Unfortunately, this algorithm is often not very effective, because its estimate has high variance.

One way of reducing variance is to use the estimator based on the following equation instead:

$$\nabla_\theta \operatorname{KL}\left[q_\theta(x) \parallel p(x|y_0)\right] = \mathbb{E}_{x \sim q_\theta(x)} \left[\left(\nabla_\theta \log(q_\theta(x)) \right) \cdot \left(\log \frac{q_\theta(x)}{p(x, y_0)} - B \right) \right] \quad (3)$$

for some constant B (called baseline or control variate). Prove the equation (3). Also, change the algorithm such that it uses this equation instead of (2). (15 marks)

- (c) Find B that minimises the variance of the estimate for $N = 1$. That is, find B that minimises

$$\operatorname{Variance} \left(\left(\nabla_\theta \log(q_\theta(x)) \right) \cdot \left(\log \frac{q_\theta(x)}{p(x, y_0)} - B \right) \right),$$

which has the same value as

$$\mathbb{E}_{x \sim q_\theta(x)} \left[\left(\left(\nabla_\theta \log(q_\theta(x)) \right) \cdot \left(\log \frac{q_\theta(x)}{p(x, y_0)} - B \right) \right)^2 \right] - \left(\nabla_\theta \text{KL} [q_\theta(x) \| p(x|y_0)] \right)^2$$

because of the equation (3).

(15 marks)